Q1

```
var myval = sc.textFile(“hdfs:/loudacre/weblogs”)
myval.map(line=>line.split(“ ”)).map(word=>(word(0),word(2)))

myval.foreach(line => println(line._1 + “/” + line._2))
```

Q2

```
myval = sc.wholeTextFiles("hdfs:/loudacre/activations");
```

myval.first(). It returns the first xml file, which contains different activations, activation time stamp, account number, device id, phone number, model.

mapvalues() can be used to map the second value to a different rdd

```
ar =myval.flatMap(lambda (filename,xmlstring): getactivations(xmlstring))
```

```
model = ar.map(lambda record: getaccount(record) + ":" + getmodel(record))
```

```
model.saveAsTextFile("/loudacre/account-models")
```

Q3
Part 1 see code.
PART 2

```
spark-submit –master local \
CountJPGs.py hdfs:/loudacre/weblogs
```

And the count is 64978

The driver program is executed on local computer, the result is stored locally

c. if submit the result over the cluster, the command is spark-submit –master yarn-client \
CountJPGs.py hdfs:/loudacre/weblogs

the driver program is executed over the cluster, and the result is stored locally.

d. Stages are operations that can run on the same data partitioning in parallel across nodes. Tasks within a stage are operations executed on one node that are pipelined together.
There is one stage, 311 tasks,

e. pipelining: when possible, spark will perform a sequence of transformations by row so no data is stored.

sc.wholeTextFiles("hdfs:/loudacre/activations").first()

we can directly access the first record in activations without storing the rest records.